**Name: Ikhwan Haziq Bin Sinwan Matrix number: 1704749I**

**What is SDLC?**

The software development life cycle (SDLC) is a framework defining tasks performed at each step in the software development process. SDLC is a structure followed by a development team within the software organization. It consists of a detailed plan describing how to develop, maintain and replace specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

SDLC consists of following activities:

1. Planning: The most important parts of software development, requirement gathering, or requirement analysis are usually done by the most skilled and experienced software engineers in the organization. After the requirements are gathered from the client, a scope document is created in which the scope of the project is determined and documented.
2. Implementation: The software engineers start writing the code according to the client's requirements.
3. Testing: This is the process of finding defects or bugs in the created software.
4. Documentation: Every step in the project is documented for future reference and for the improvement of the software in the development process. The design documentation may include writing the application programming interface (API).
5. Deployment and maintenance: The software is deployed after it has been approved for release.
6. Maintaining: Software maintenance is done for future reference. Software improvement and new requirements (change requests) can take longer than the time needed to create the initial development of the software.

SDLC works by lowering the cost of software development while simultaneously improving quality and shortening production time. SDLC achieves these apparently divergent goals by following a plan that removes the typical pitfalls to software development projects. That plan starts by evaluating existing systems for deficiencies. Next, it defines the requirements of the new system. It then creates the software through the stages of design, development, testing, and deployment. By anticipating costly mistakes like failing to ask the end user for suggestions, SLDC can eliminate redundant rework and after-the-fact fixes.

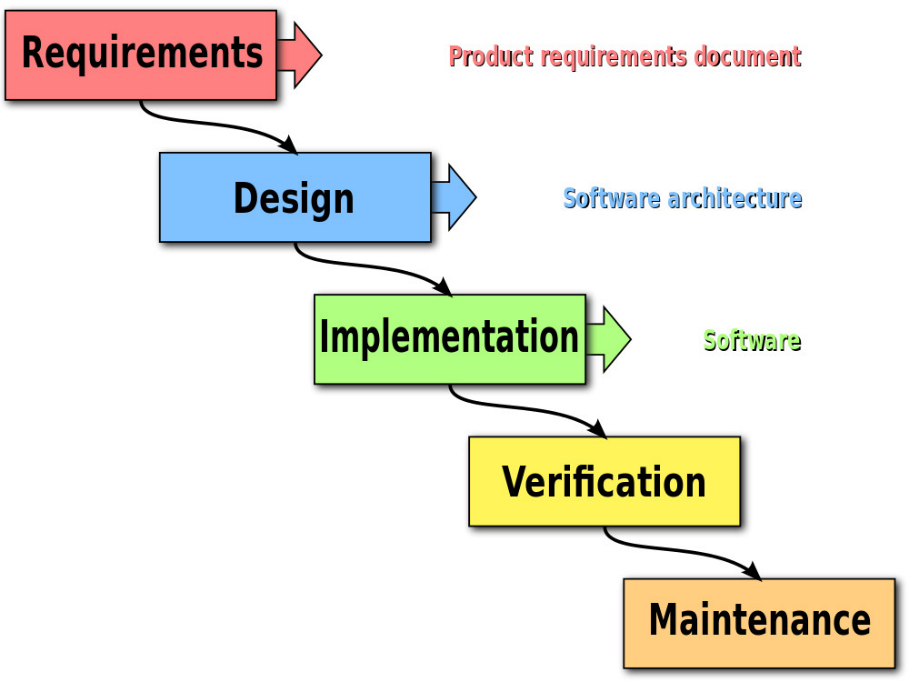
**3 models**

**Waterfall Model**

This SDLC model is the oldest and most straightforward. With this methodology, we finish one phase and then start the next. Each phase has its own mini-plan and each phase “waterfalls” into the next. The biggest drawback of this model is that small details left incomplete can hold up the entire process.

The advantage of waterfall development is that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one. The waterfall model progresses through easily understandable and explainable phases and thus it is easy to use. It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process. In this model, phases are processed and completed one at a time and they do not overlap. Waterfall model works well for smaller projects where requirements are very well understood.

The Disadvantage of waterfall development is that it is difficult to estimate time and cost for each phase of the development process. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-thought out in the concept stage. Not a good model for complex and object-oriented projects. Not suitable for the projects where requirements are at a moderate to high risk of changing.

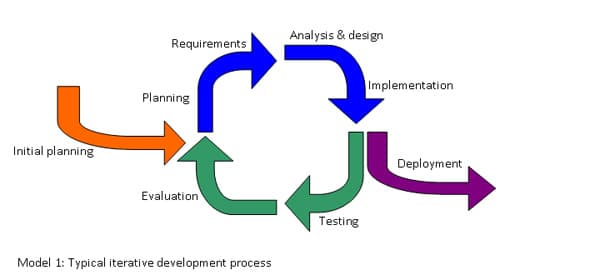


**Iterative Model**

This SDLC model emphasizes repetition. Developers create a version very quickly and for relatively little cost, then test and improve it through rapid and successive versions. One big disadvantage here is that it can eat up resources fast if left unchecked.

The advantage of iterative model is that it Generates working software quickly and early during the software life cycle. More flexible – less costly to change scope and requirements. Easier to test and debug during a smaller iteration. Easier to manage risk because risky pieces are identified and handled during its iteration. Each iteration is an easily managed milestone.

The advantage of iterative model is that each phase of an iteration is rigid and do not overlap each other. Problems may arise pertaining to system architecture because not all requirements are gathered up front for the entire software life cycle.



**Big Bang Model**

This high-risk SDLC model throws most of its resources at development and works best for small projects. It lacks the thorough requirements definition stage of the other methods. The Big Bang model is an SDLC model where we do not follow any specific process. The development just starts with the required money and efforts as the input, and the output is the software developed which may or may not be as per customer requirement. This Big Bang Model does not follow a process/procedure and there is a very little planning required. Even the customer is not sure about what exactly he wants, and the requirements are implemented on the fly without much analysis.

The Big Bang Model comprises of focusing all the possible resources in the software development and coding, with very little or no planning. The requirements are understood and implemented as they come. Any changes required may or may not need to revamp the complete software.

This model is ideal for small projects with one or two developers working together and is also useful for academic or practice projects. It is an ideal model for the product where requirements are not well understood, and the final release date is not given.

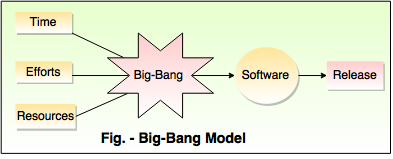
The advantage of this Big Bang Model is that it is very simple and requires very little or no planning. Easy to manage and no formal procedure are required.

However, the Big Bang Model is a very high-risk model and changes in the requirements or misunderstood requirements may even lead to complete reversal or scraping of the project. It is ideal for repetitive or small projects with minimum risks.

The advantages of the Big Bang Model are as follows −

* This is a very simple model
* Little or no planning required
* Easy to manage
* Very few resources required
* Gives flexibility to developers
* It is a good learning aid for new comers or students.

The advantages of the Big Bang Model are as follows –

* Very High risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Can turn out to be very expensive if requirements are misunderstood.

<https://www.techopedia.com/definition/22193/software-development-life-cycle-sdlc>

<https://stackify.com/what-is-sdlc/>

<https://www.testingexcellence.com/iterative-model/>

<http://toolsqa.com/software-testing/waterfall-model/>

<https://www.tutorialspoint.com/sdlc/sdlc_bigbang_model.htm>